COMPLEX EVENT PROCESSING
(ENodeB User-Plane Aggregation)

By

Ayush Choubey

Acknowledgment

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. I give a special gratitude to our project manager Rakesh Sir who gave us such an awesome project. I would also thank Jatin sir, who helped me in the project from the very start by briefing about the project and suggesting the important changes, to make the program perform even better.

Furthermore, I would also like to thanks to Lavanya ma'am who helped me BER Decoder API's and helping in improving the abstraction of the program.

Lastly, i would like to appreciate the help of my fellow intern Simar for being there and giving me an introduction to machine learning.

Abstract

Event Processing is a method of analyzing and processing stream of information(data) about anything that happen(event). Complex Event Processing in the processing of information from many sources and combining them to infer something.

In my prototype, the source were many ENodeB data's. The events were any operation related to mobile(like calling, data uplink and downlink volume etc). These events were represented in the form of CDR's which are streamed continously to the consolidator.

ENodeB or eNB is that part of the LTE network that directly deals with the mobile handsets,i.e, it is the component that receives the entire user information directly from the handset.

The User-Plane aggregation deals with the aggregation of ENodeB data, so that it can be used for other purposes like Online Analysis etc.

The project uses Storm(Apache licensed) created by NathanMarz which is a distributed and fault-tolerant realtime computation system. It's highly scalable and gurantees the message processing and thus was perfect for the project.

The call data records generated(after every 1.28 seconds) and are streamed contiously and can be aggregated after a certain time interval(say 1minute) in the consolidator. Now using the capabilities of storm, we are free to either stream the aggregated record(for some analysis), store it into database, generate output files, etc.

Table of Contents

Introduction

In the industry its very important to have a scalable, fast, reliable and powerful system. These factors becomes all the more important when we have to deal with big data i.e, is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.

In the current implementation, this kind of Big data were handled by File Event, in which the entire aggregation logic was written in DUP. Though fast, scalable and reliable, but the system still lacked the power that would have been there, had other languages like java would have been used.

My project as an intern at ericsson was to develop a prototype, that would have given the atleast the same speed, scalability and reliability as the present FE and the power that java developers can have.

The main Objective of the project was to :

- Use Storm as the computation system and see if the prototype developed on it is comparable to that of the Current implementation of FE

Company Profile

Ericsson is a Swedish multinational technology company that provides and operates telecommunications networks, television and video systems, and related services. It was founded by Lars Magnus Ericsson in 1876 at stockholm, sweden which is the current global headquarters also. From that time, the company has made immense progress by expanding itself to more than 180 countries with 1,27,967 employees(as of 2012).

Ericsson offers products and services from three main business units :

1) **BNET** : Bussiness Unit Network which focus on etworks for mobile phones and fixed line public telephone networks
2) **BUGS** : Bussiness unit Global Services, which provides telecoms-related professional services, including for example taking responsibility for running an operators network and related business support systems.
3) **BUSS** : Business Unit Support Solutions focuses on Operations support systems/Business support systems(OSS/BSS), TV and media and Mobile commerce(M-Commerce).
4) **Research and Development** : Research and development department deals with the research of new technologies that can be used by the ericsson software.

“In the rapidly changing information communications industry, we take the role of business enabler”

-Hans Vestberg

(President & CEO)

Methodology

The entire project is built over Apache Storm, which is a distributed, fault tolerant computation system. It is highly scalable and gurantees that message processing.

Storm is a multinode system and has two kind of nodes :
Master Node and Worker Node.

Master Node runs a daemon called Nimbus and this node deals with code distribution around the cluster, assigning tasks to machines, and monitoring for failures.

Worker node runs a daemon called Supervisor and this node actually does all kind of processing job and executes the topology.

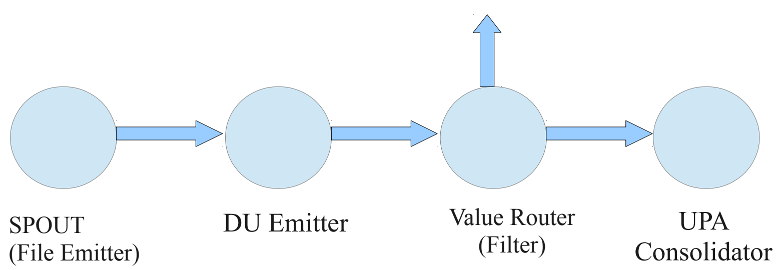
All the coordination between nimbus and worker is done by the help of zookeeper cluster.

- Each Worker node runs one or more worker process.
- Each Worker Process runs one or more Executors
- Each Executor runs one or more task.
- Each task can either be a spout or a Bolt. A spout is data generation component of Storm and a Bolt is a data Processing component of Storm.

Other than Storm, I used ASNParser and BerDecoder that were written in java for some older versions of Multi-Mediation. A few minor changes were made in them so as to make it usable for our project.

Prototype Architecture

The architecture of prototype was quite simple and none of the component were actually totally dependent on each other, so that future changes can be incorporated quite easily.



The data(CDR or Call Data Record) from the ENodeB were encoded in BER Format. Each BER File were named by following a particular pattern that contained the timestamp of the file, ENodeB Id and the file sequence number.

The Spout or File Emitter job was to loop continously, find a ENodeB file and send it to a particular DUEmitter according to the ENodeB Id given in the filename.

A DUEmitter thread always receives a file of a particular ENodeB. Since the files are BER Encoded, they are first decoded and is stored in a DataStructure called JDataUnit. Each JDataUnit or JDU contains one CDR. As soon as we get a JDU, we emit it to Value Router.

Each Thread of ValueRouter receives a JDU from a particular DUEmitter. The value Router then routes the DU to two streams according to the event header :-

- toUPAConsolidator
- toFormatter

The CDR's which are important for User Plane Aggregation are routed using toUPAConsolidator stream and the one's required by Formatter for other modules are routed using toFormatter Stream.

Each Thread Of UPA Consolidator receives a CDR(JDU) from a particular ValueRouter thread, that means DU's of a particular ENodeB Id always go to a particular thread of UPA Consolidator.

In UPA Consolidator the CDR's are merged continously until they run out of time window(currently set as 1 minute).

Result and Analysis

The Following are the test result when the prototype was tested with a 16 core machine with hyperthreading(32 virtual cores) and 64 GB RAM.

For Decoding BER Files

While decoding a BER File with 57,156 CDR it took about 38 seconds

For End to End Processing

Number of CDRs	Time Taken	Number Of ENodeB	Speed
858200	3 min	5	4672 CDR/sec
1716400	3 min 33 secs	10	8058 CDR/sec

The 2nd output with 1716400 CDR was produced using 30% CPU and 16 Gigs of RAM

After analysis of the result we found that, the greater the number of ENodeB, greater will be the parallelism and hence greater will be the speed.

We also found that, increasing the number of process didn't actually made any significant difference in the output.

By using VisualVm profiler, we found that most of the time was actually spent on decoding rather than aggregation.

Conclusion

Prototype was quite successful. The result showed that, it was scalable, reliable and was very powerful. The prototype used about one-third of the entire processing power and gave about 8000+ CDR per second.

On the basis of the prototype and seeing the analysis of the results we can safely assume that if we increase the scaling and use total capability of a system, we can get even more than 25,000 CDR per Sec on a single node.

On the basis of Results, we can conclude that we can migrate from the DUP implementation of FE to Java Implementation to get better scalability and power.

Scope Of Further Expansion

There are many more Use-Cases, where this project can be used, few of them are

1) Aggregation of MME(Mobility Management Entity) data
2) Handover Case
3) Aggregation Of ENodeB, MME and SAE data

Aggregation of MME data :

Like ENodeB, MME is also a very important part of LTE network and data from it also contains very important information. Aggregating all those information can also provide very important statistics.

Handover Cases :

In normal scenarios, a user subscribes to a particular ENodeB, bur sometimes during migration ENodeB changes and those cases should also be handled. Aggregating all those data about the UE's that have been migrated can prove quite vital for the the telecom companies

Aggregation Of EnodeB , MME and SAE data

In LTE networks, a UE is not just used for calling but for internet surfing video calling etc. In this Use case, information from all these three data sources are aggregated, which will give a better stats of the User's usage pattern

Use Of Aggregated Data

The aggregated data can further be used to acquire necessary statistical information which can be used for various other purposes like generating special offers for the user for particular type of internet usage, etc.

All the aggregated data can further be used with MOA Framework and provide better and automated analysis of the results.

Appendices

User Plane Session Correlation logic

Category	Name	Type	Value	Source event and fields	Aggregation	Unit	in L11A
ID	start	NUMERIC(18,6)	start time of this UP session	INTERNAL_PER_UE_TRAFFIC_REP start - 1.28 sec	-	sec	Y
	duration	NUMERIC(18,6)	aggregation period		-	sec	Y
	ts_offset	FLOAT			-	sec	Y
	enbs1apid	BIGINT	The eNB UE S1AP ID used for the UE (defined in TS 36.413)	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_ENBS1APID	-		Y
	mme1apid	BIGINT	The MME UE S1AP ID used for the UE (defined in TS 36.413)	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_MMES1APID	-		Y
	gummei	TEXT	The globally unique MME identifier, consists of PLMN id (3 bytes), MME Group ID (2 bytes), MME code (1 byte). (defined in TS 36.413)	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_GUMMEI	-		Y
	rac_ue_ref	INT	eNb local UE identifier	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_RAC_UE_REF	-		Y
	cell_id	INT	eNb internal cell identity, unique in the eNb.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_CELL_ID	-		Y
	enb_name	TEXT	eNb node id (PLMN + node id + cell id = unique cell id)	INTERNAL_PER_UE_TRAFFIC_REP - enb_name	-		Y
THROUGHPUT	vol_thp_ul	INT	Number of transmitted bytes (PDCP level) used for throughput calculation in the uplink (truncated with last TTIs).	INTERNAL_PER_UE_TRAFFIC_REP, INTERNAL_PER_UE_RB_TRAFFIC_REPORT - EVENT_PARAM_PER_PDCPVOL_UL_RB - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_UL	\sum{(\sum{EVENT_PARAM_PER_PDCPVOL_UE_RB} - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_UL)}	byte	Y
	vol_thp_dl	INT	Number of transmitted bytes (PDCP level) used for throughput calculation in the downlink (truncated with the last TTIs).	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PDCP_DRB_ACKVOL_DL - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_DL	\sum{(EVENT_PARAM_PER_UE_PDCP_DRB_ACKVOL_DL- EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_DL)}	byte	Y
	vol_total_ul	INT	VOL_THP_UL + Number of bytes transmitted (PDCP level) in the last TTIs in the uplink.	INTERNAL_PER_UE_TRAFFIC_REP, INTERNAL_PER_UE_RB_TRAFFIC_REPORT - EVENT_PARAM_PER_PDCPVOL_UL_RB - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_UL	**SUM**	byte	Y
	vol_total_dl	INT	VOL_THP_DL + Number of bytes transmitted (PDCP level) in the last TTIs in the downlink.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PDCP_DRB_ACKVOL_DL - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_DL	**SUM**	byte	Y
	time_thp_ul	INT	Number of ms used for the throughput calculation in uplink.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_THP_TIME_UL	**SUM**	sec	Y
	time_thp_dl	INT	Number of ms used for the throughput calculation in downlink	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_THP_TIME_DL	**SUM**	sec	Y
	thp_ave_ul	REAL	Average throughput in the uplink calculated over the session length.	INTERNAL_PER_UE_TRAFFIC_REP, INTERNAL_PER_UE_RB_TRAFFIC_REPORT - EVENT_PARAM_PER_PDCPVOL_UL_RB - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_UL - EVENT_PARAM_PER_UE_THP_TIME_UL	**SUM**(8*(**SUM** (EVENT_PARAM_PER_PDCPVOL_UE_RB) - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_UL) **/** **SUM**(EVENT_PARAM_PER_UE_THP_TIME_UL)	bps	Y
	thp_ave_dl	REAL	Average throughput in the downlink calculated over the session length.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PDCP_DRB_ACKVOL_DL - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_DL - EVENT_PARAM_PER_UE_THP_TIME_DL	**8*SUM**(EVENT_PARAM_PER_UE_PDCP_DRB_ACKVOL_DL- EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_DL) **/** **SUM**(EVENT_PARAM_PER_UE_THP_TIME_DL)	bps	Y
	thp_detailed_ul	REAL[]	Set of 1.28 sec throughput samples in the uplink during the session length.	INTERNAL_PER_UE_TRAFFIC_REP, INTERNAL_PER_UE_RB_TRAFFIC_REPORT - EVENT_PARAM_PER_PDCPVOL_UL_RB - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_UL - EVENT_PARAM_PER_UE_THP_TIME_UL	**ARRAY**	bps	Y
	thp_detailed_dl	REAL[]	Set of 1.28 sec throughput samples in the downlink during the session length.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PDCP_DRB_ACKVOL_DL - EVENT_PARAM_PER_UE_THP_PDCPVOL_TRUNK_DL - EVENT_PARAM_PER_UE_THP_TIME_DL	**ARRAY**	bps	Y
	samples_ul	INT	Number of 1.28 sec throughput data samples reported in the uplink (i.e., with non-zero data volume).	INTERNAL_PER_UE_TRAFFIC_REP	**SUM**		Y
	samples_dl	INT	Number of 1.28 sec throughput data samples reported in the downlink (i.e., with non-zero data volume).	INTERNAL_PER_UE_TRAFFIC_REP	**SUM**		Y
LATENCY & DELAY	latency_dl	REAL	DL latency measure. The time is between reception of a PDCP SDU untill the first succesful transmission of a MAC SDU have been transmitted	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_LAT_TIME_DL - EVENT_PARAM_PER_UE_LAT_SAMPL_DL	**SUM**(EVENT_PARAM_PER_UE_LAT_TIME_DL) **/** **SUM**(EVENT_PARAM_PER_UE_LAT_SAMPL_DL)	ms	Y
	dl_rlc_delay	REAL	Downlink delay measure in the RLC layer. One sample per RLC SDU sent to MAC. The time for each sample is between reception of a packet (PDCP SDU) until the packet is sent to the MAC layer entity for transmission on the air.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_DL_RLC_DELAY - EVENT_PARAM_PER_UE_DL_RLC_DELAY_SAMPL_DL	**SUM**(EVENT_PARAM_PER_UE_DL_RLC_DELAY) **/** **SUM**(EVENT_PARAM_PER_UE_DL_RLC_DELAY_SAMPL_DL)	ms	Y
	dl_mac_delay	REAL	Downlink delay measure in the MAC layer. One sample per MAC SDU. The time for each sample is between reception of a packet RLC PDU until the packet is received by the UE or HARQ failure on this MAC SDU.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_DL_MAC_DELAY - EVENT_PARAM_PER_UE_DL_RLC_DELAY_SAMPL_DL	**SUM**(EVENT_PARAM_PER_UE_DL_MAC_DELAY) **/** **SUM**(EVENT_PARAM_PER_UE_DL_RLC_DELAY_SAMPL_DL)	ms	Y
	delay_samples	INT	Number of samples to be used in delay measure calculations	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_DL_RLC_DELAY_SAMPL_DL	**SUM**		Y
SCHEDULING	sched_restrict_ue_cat_ul	INT	The sum of the contributions from each UE of the number of ms where respective UE is limited in the DL direction by its UE capability.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_SCHED_RESTRICT_UE_CAT_UL	**SUM**	ms	N
	sched_restrict_ue_cat_dl	INT	The sum of the contributions from each UE of the number of ms where respective UE is limited in the UL direction by its UE capability.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_SCHED_RESTRICT_UE_CAT_DL	**SUM**	ms	N
	sched_activity_ue_ul	INT	The sum of the contributions from each UE in the cell of the number of ms where SRB/DRB data has been scheduled in the UL.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_SCHED_ACTIVITY_UE_UL	**SUM**	ms	Y
	sched_activity_ue_dl	INT	The sum of the contributions from each UE in the cell of the number of ms where SRB/DRB data has been scheduled in the DL.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_SCHED_ACTIVITY_UE_DL	**SUM**	ms	Y

PACKET	packet_tr_dl	INT	Number of PDCP SDUs transmitted DL	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PACKET_TR_DL	**SUM**		Y
	packet_rec_dl	INT	Number of PDCP SDUs received DL	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PACKET_REC_DL	**SUM**		Y
	packet_lost_ho_dl	INT	Number of PDCP SDUs lost due to handover, DL	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PACKET_LOST_HO_DL	**SUM**		Y
	packet_lost_pelr_dl	INT	Number of PDCP SDUs lost due to Traffic Management, DL	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PACKET_LOST_PELR_DL	**SUM**		Y
	packet_rec_ul	INT	Number of PDCP SDUs received, UL	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PACKET_REC_UL	**SUM**		Y
	packet_lost_ul	INT	Number of PDCP SDUs lost, UL	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_PACKET_LOST_UL	**SUM**		Y
RLC	rlc_ack_dl	INT	The total number of successful RLC PDU transmissions (ACKs) in the downlink direction.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_RLC_ACK_DL	**SUM**		Y
	rlc_nack_dl	INT	The total number of unsuccessful RLC PDU and RLC PDU segment transmissions (NACKs) in the downlink direction.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_RLC_NACK_DL	**SUM**		Y
	rlc_ack_ul	INT	The total number of successful RLC PDU transmissions (ACKs) in the uplink direction.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_RLC_ACK_UL	**SUM**		Y
	rlc_nack_ul	INT	The total number of unsuccessful RLC PDU and RLC PDU segment transmissions (NACKs) in the uplink direction.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_RLC_NACK_UL	**SUM**		Y
HARQ	harq_dl_ack_qpsk	INT	The total number of successful HARQ transmissions in the downlink direction using a QPSK modulation. Successful is based on the HARQ ACK from the UE.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_DL_ACK_QPSK	**SUM**		Y
	harq_dl_nack_qpsk	INT	The total number of unsuccessful HARQ transmissions in the downlink direction using a QPSK modulation. Failure is based on the HARQ NACK from the UE.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_DL_NACK_QPSK	**SUM**		Y
	harq_ul_succ_qpsk	INT	The total number of successful HARQ transmissions in the uplink direction using a QPSK modulation. Successful is based on the CRC check, not based on if RBS sends HARQ ACK (RBS can use the ACK even if the transport block was not successfully decoded in a way to control the HARQ)	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_UL_SUCC_QPSK	**SUM**		Y
	harq_ul_fail_qpsk	INT	The total number of unsuccessful HARQ transmissions in the uplink direction using a QPSK modulation. Failure is based on the CRC check (which will result in an NACK).	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_UL_FAIL_QPSK	**SUM**		Y
	harq_dl_ack_16qam	INT	The total number of successful HARQ transmissions in the downlink direction using a 16QAM modulation. Successful is based on the HARQ ACK from the UE.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_DL_ACK_16QAM	**SUM**		Y
	harq_dl_nack_16qam	INT	The total number of unsuccessful HARQ transmissions in the downlink direction using a 16QAM modulation. Failure is based on the HARQ NACK from the UE.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_DL_NACK_16QAM	**SUM**		Y
	harq_ul_succ_16qam	INT	The total number of successful HARQ transmissions in the uplink direction using a 16QAM modulation. Successful is based on the CRC check, not based on if RBS sends HARQ ACK (RBS can use the ACK even if the transport block was not successfully decoded in a way to control the HARQ)	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_UL_SUCC_16QAM	**SUM**		Y
	harq_ul_fail_16qam	INT	The total number of unsuccessful HARQ transmissions in the uplink direction using a 16QAM modulation. Failure is based on the CRC check (which will result in an NACK).	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_UL_FAIL_16QAM	**SUM**		Y
	harq_dl_ack_64qam	INT	The total number of successful HARQ transmissions in the downlink direction using a 64QAM modulation. Successful is based on the HARQ ACK from the UE.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_DL_ACK_64QAM	**SUM**		Y
	harq_dl_nack_64qam	INT	The total number of unsuccessful HARQ transmissions in the downlink direction using a 64QAM modulation. Failure is based on the HARQ NACK from the UE.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_UE_HARQ_DL_NACK_64QAM	**SUM**		Y
	mac_dtx_ul_qpsk	INT	The total number of occasions when an uplink grant was meant for HARQ transmission in the uplink direction with QPSK, where DTX is considered the reason for no reception of HARQ in uplink in the eNB.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_MAC_DTX_UL_QPSK	**SUM**		Y
	mac_dtx_dl_qpsk	INT	The total number of occasions when an downlink HARQ feedback was not received from an UE for a Transport Block with QPSK and DTX is considered the reason.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_MAC_DTX_DL_QPSK	**SUM**		Y
	mac_dtx_dl_16qam	INT	The total number of occasions when an uplink grant was meant for HARQ transmission in the uplink direction with 16QAM, where DTX is considered the reason for no reception of HARQ in uplink in the eNB.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_MAC_DTX_DL_16QAM	**SUM**		Y
	mac_dtx_ul_16qam	INT	The total number of occasions when an downlink HARQ feedback was not received from an UE for a Transport Block with 16QAM and DTX is considered the reason.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_MAC_DTX_UL_16QAM	**SUM**		Y
	mac_dtx_dl_64qam	INT	The total number of occasions when an uplink grant was meant for HARQ transmission in the uplink direction with 64QAM where DTX is considered the reason for no reception of HARQ in uplink in the eNB.	INTERNAL_PER_UE_TRAFFIC_REP - EVENT_PARAM_PER_UE_MAC_DTX_DL_64QAM	**SUM**		Y
RADIO THROUGHPUT	radiothp_vol_dl	INT	The total successfully transferred data volume on MAC level in the downlink. This counter includes possible padding bits.	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_RADIOTHP_VOL_DL	**SUM**	byte	Y
	radiothp_res_dl	INT	The total amount of physical resources used for transmission in the downlink. Both successful and unsuccessful transmissions are counted.	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_RADIOTHP_RES_DL	**SUM**	resource element	Y
	radiothp_vol_ul	INT	The total successfully transferred data volume on MAC level in the uplink. This counter includes possible padding bits.	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_RADIOTHP_VOL_UL	**SUM**	byte	Y
	radiothp_res_ul	INT	The total amount of physical resources used for transmission in the uplink. Both successful and unsuccessful transmissions are counted.	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_RADIOTHP_RES_UL	**SUM**	resource element	Y
CQI	cqi_med	INT[16]	Set of CQI median values during the session length	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_CQI_REPORTED_XX	**ARRAY**		N
RANK & TX_MODE	ch_rank_rep	INT[2]	Number of instances of channel rank=1,2 reported (in spatial multiplexing mode), sum of the corresponding 1.28 sec counters over the session length	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_RANK_REPORTED_X	**PDF**		N
	tx_mode	INT[5]	Number of instances of tx mode=0,1,2,3,4 used, sum of the corresponding 1.28 sec counters over the session length	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_RANK_TX_X	**PDF**		N
POWER	power_restrict	INT	The number of Transport Blocks on MAC level scheduled in uplink where the UE was considered to be power limited.	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_TBSPWRRESTRICTED	**SUM**		Y
	power_no_restrict	INT	The number of Transport Blocks on MAC level scheduled in uplink where the UE was NOT considered to be power limited.	INTERNAL_PER_RADIO_UE_MEASUREMENT - EVENT_PARAM_TBSPWRUNRESTRICTED	**SUM**		Y
PERIODIC RADIO MEASUREMENTS	rsrp_avg	REAL	Average of RSRP measurements during the aggregation period	UE_MEAS_INTRAFREQ1/UE_MEAS_INTRAFREQ2 -rsrpserving	**AVG(rsrpserving values)**		Y
	rsrq_avg	REAL	Average of RSRQ measurements during the aggregation period	UE_MEAS_INTRAFREQ1/UE_MEAS_INTRAFREQ2 -rsrqserving	**AVG(rsrqserving values)**		Y
	rsrp_detailed	REAL[]	List of RSRP measurements during the aggregation period	UE_MEAS_INTRAFREQ1/UE_MEAS_INTRAFREQ2 -rsrpserving	**ARRAY**		Y
	rsrq_detailed	REAL[]	List of RSRQ measurements during the aggregation period	UE_MEAS_INTRAFREQ1/UE_MEAS_INTRAFREQ2 -rsrqserving	**ARRAY**		Y
	ue_meas_ts	FLOAT_16[]	List of timestamps of the RSRP/RSRQ measurements	UE_MEAS_INTRAFREQ1/UE_MEAS_INTRAFREQ2 -event timestamps	**ARRAY**		Y

Bibliography

- **Getting started with Storm** By *Jonathan Leibuisky, Gabriel Eisbruch & Dario Simonassi*
- **Storm-User Google groups** :-
  - https://groups.google.com/forum/#!forum/storm-user
- **Storm tutorials By Michael G. Noll** :-
  - http://www.michael-noll.com/blog/2012/10/16/understanding-the-parallelism-of-a-storm-topology/
- **A layman guide to subset of ASN.1,BER, DER :-**
  - http://luca.ntop.org/Teaching/Appunti/asn1.html
- Previous Implementation of User Plane Aggregation in DUP